Craig Carlson
March 24th, 2014
Web Applications

Hashing in Ruby on Rials

Hashes are a great way to store large amounts of data, and since rails is a great way store large amounts of data. This is why Google stores there large scale data in hashes. To begin with hashing in ruby is a collection of key value pairs and is like an array in that there are indexes except that in a hash the keys are arbitrary keys of any object type, not an integer index. This allows for the hash to contain any object type because the hash can be created on any type.
There are two different ways to initialize a hash:

```
months = Hash.new( "month" )

or

months = Hash.new "month"
```

This creates the hash with the default values of the "month" so that if a hash value if accessed that doesn't exist then the default value will be returned.

In order to get and add values to the hash tables:

```
H = Hash["a" => 100, "b" => 200]

puts "#{H['a']}"
puts "#{H['b']}"
```

Hash also has built in methods that can only be called after an instance of the hash has been created. I order to create this new instance of the hash:

```
Hash[[key =>|, value]* ] or

Hash.new [or] Hash.new(obj) [or]

Hash.new { |hash, key| block }
```

Once we have this new instance of a hash we are able to get the keys of the hash returned as a list:

keys = months.keys

puts "#{keys}"

There are several other hash methods that are allowed to be called once an instance of the hash has been created.
other_hash: There is the other_hash that allows you to compare two hashes and see if they are equal at an index level.
Hash.[key] – this allows you to, using a key, reference values from the hash. If the hash is not found then the default value is returned.
-You are also able to compare this value that is returned from the hash to another variable that is not associated with the hash.
Hash.clear: clears all of the key value pairs from the hash table
You are also able to set the default value of the hash after it has been initialized for a specific key and if that key does not exist then it will use a given object.
Hash.default_proc: this returns a block of code if the hash was created by a block

We are also able to delete specific values using the key.
Hash.each{|key,value| block} This allows for us to iterate over the entire hash and do operations like you would with any array.
The **hash.fetch** allows you to fetch a certain value based on the a key inconstant time which is potentially much faster that any array because at a large scale we see huge time discrepancies.
There are also lots of functions to manipulate the hash and find and reverse indexes.

Sources:
http://www.tutorialspoint.com/ruby/ruby_hashes.htm
http://api.rubyonrails.org/classes/Hash.html
http://www.ruby-doc.org/core-2.1.1/Hash.html
https://code.google.com/p/sparsehash/